Electronic Version
Stylesheet Version v1.1.1

Description

METHOD AND RELATED APPARATUS FOR CLEARING DATA IN A MEMORY

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to a method and related apparatus for clearing data in a memory, and more particularly, to a method and related apparatus for clearing data in a memory without the involvement of a CPU.

[0003] 2. Description of the Prior Art

[0004] Fig.1 is a schematic diagram of a conventional computer system 10. As shown in Fig.1, the computer system 10 includes a CPU 12, a north bridge circuit 14, a south bridge circuit 16, a display controller 18, a display 19, a memory 20, a hard disk 22, and an input device 24. The memory 20 includes a plurality of memory units 26 arranged in arrays, i.e., each memory unit 26 corresponds to a column address and a row address. The accessing operations of

the memory 20 are controlled via a memory controller 30 positioned in the north bridge circuit 14. The memory controller 30 includes an address register 32 and a data register 34 where the address register 32 is for storing memory addresses, and the data register 34 is for storing data to be written to the memory 20 and data read from the memory 20.

[0005] For the computer system 10, any executed programs, for example a driver or an application program, require the memory 20 for storing data. When a first application program is executed, a memory block of the memory 20 is designated for storing operation data of the first application program. While the first application program is closed, the memory block must be released so that other programs can use this memory block to store data. In addition to release the memory block, however, the first application program must clear the memory block, by for example overwriting the data stored in each memory unit 26 of the memory block with a logic value "1" or "0". In such case, a second application program executed thereafter, can correctly access data in the same memory block. If that memory block is not cleared, errors due to misjudgment may occur when the second application pro-

gram is executed. These errors may even lead to the crash of the computer system 10.

[0006] Therefore, when a program requires a certain capacity of memory units 26 in the memory 20 to store operation data, that certain capacity of memory units 26 must be overwritten with logic values "1" or "0". For example, if the CPU 12 executes the program codes of clearing data, the CPU will output the memory addresses corresponding to the memory units 26 to be used to the address register 32. Meanwhile, the CPU 12 will repeatedly output the logic value "1" or "0" to the data register 34. If the capacity of the memory 20 to be used is 3MB, the CPU 12 will output the logic values "1" to the data register 34 24 million times for clearing 24 million memory units 26 (corresponding to 3MB capacity) in the memory 20. It can be seen that the CPU 12 spends much time repeatedly outputting the logic value "1" or "0" to the memory 20, thereby reducing its efficiency. In addition to reducing the efficiency of the CPU 12, the limited bandwidth of the front-side bus (FSB) between the CPU 12 and the north bridge circuit 14 is also consumed. This also affects the total efficiency of the computer system 10.

SUMMARY OF INVENTION

[0007] It is therefore a primary objective of the present invention to provide a method and related apparatus of clearing data in a memory for solving the above problems.

[0008] According to the claimed invention, a method of clearing data in a memory of a computer system is disclosed. The computer system includes a processor, and a memory controller electrically connected to the processor and the memory for controlling the accessing operations of the memory. The method includes the processor generating a predetermined logic value and delivering the predetermined logic value to the memory controller, and the memory controller repeatedly overwriting data stored in the plurality of memory units by the predetermined logic value.

[0009] The claimed invention further includes a computer system including a processor for controlling operations of the computer system, a memory having a plurality of memory units for respectively storing a data, and a memory controller electrically connected to the processor and the memory. The memory controller includes an address register for storing a plurality of memory addresses corresponding to the plurality of memory units, a data register, and a data clear module for transmitting a predetermined

logic value generated by the processor to the data register so that the predetermined logic value overwrites data stored in the plurality of memory units one by one.

[0010] These and other objectives of the present invention will no doubt become obvious to those of ordinary skill in the art after having read the following detailed description of the preferred embodiment that is illustrated in the various figures and drawings.

Brief Description of Drawings

[0011] Fig.1 is a schematic diagram of a conventional computer system 10.

[0012] Fig.2 is a schematic diagram of a computer system according to a first embodiment of the present invention.

[0013] Fig.3 is a memory address table of the present invention.

[0014] Fig.4 is a schematic diagram of another computer system according to a second embodiment of the present invention.

Detailed Description

[0015] Fig.2 is a schematic diagram of a computer system 80 according to a first preferred embodiment of the present invention. As shown in Fig.2, the computer system 80 includes a CPU 82, a north bridge circuit 84, a south bridge

circuit 86, a display controller 88, a memory 90, an input device 92, a hard disk 94, and a display 96. The north bridge circuit 84 has a memory controller 98, and the memory controller 98 includes a data clear module 100, an address register 102, and a data register 104. The memory 90 includes a plurality of memory units 106 arranged in arrays, i.e., each memory unit 106 corresponds to a column address and a row address. The memory controller 98 is for controlling accessing operations of the memory 90. The address register 102 is for storing memory addresses, and the data register 104 is for storing data to be written to the memory 90 or data to be read from the memory 90. The data clear module 100 can output a predetermined logic value (such as "1" or "0") to overwrite the memory units 106 of the memory 90 so as to clear any data in the memory units 106. Therefore, the memory controller 98 is capable of deleting data stored in the memory in virtue of the installation of the data clear module 100.

[0016] The mechanism of the data clear module 100 is demonstrated as follows. When the computer system 80 starts, an operating system is loaded first. Then the user inputs an instruction via the input device 92 for executing an ap-

plication program. The application program will be allotted a portion of the memory units 106 of the memory 90 for storing operation data. Those memory units 106 used for storing operation data must be cleared by the application program to prevent operation errors. Therefore, the CPU 82 delivers a control instruction to the memory controller 98 for starting the data clear module 100. In addition, the CPU 82 also delivers the memory addresses corresponding to the memory units 106 to be cleared to the address register 102. Accordingly, the data clear module 100 can select the logic value "1" or "0" to overwrite data stored in the portion of the memory units 106.

[0017] As described, instead of outputting the logic value repeatedly to the data register 104, the logic value is generated by the CPU 82 and only has to be delivered to the data clear module 100 once when clearing data. Therefore, the efficiency of the CPU 82 is improved. In addition, since the CPU 82 does not need to repeatedly deliver the logic value to the data register 104, it only has to directly output the logic value to the data clear module 100 once. The bandwidth of the FSB between the CPU 82 and the north bridge circuit 84 is not consumed.

[0018] In addition, if the data to be cleared includes a plurality of

data bits, the memory controller 98 generally uses physical memory addresses (for example a memory address table) to access memory units 106 of the memory 90. Fig.3 is a schematic diagram of a memory address table 107. As shown in Fig.3, the memory address table 107 includes three kinds of fields where fields 108a, 108b, and 108n record the physical memory addresses, fields 110a, 110b, and 110n record flags which represent whether the data is an end portion (end of file, EOF), and fields 112a, 112b, 112n designate a bit length of each physical memory address recorded in fields 108.

[0019] For example, if a program (an application or a driver) has to clear the data stored in a plurality of memory units 106 in the memory 90, the program first generates the memory address table 107 via the operation system of the computer system 80, and stores the memory address table 107 in a predetermined memory block of the memory 90. Then, the CPU 82 executes the program codes of the program, and thus outputs a control instruction to the memory controller 98 so as to start the data clear module 100. While being started, the data clear module 100 reads the memory address table 107 in order to distinguish exactly the physical memory addresses to be cleared. Con-

sequently, the data clear module 100 can read a memory address ADDRESSa recorded in field 108a, and consecutively clear a plurality of data bits from the memory address ADDRESSa according to a bit length LENGTHa recorded in field 112a. In other words, the physical memory addresses corresponding to the plurality of data bits are consecutively written to the address register 102 (e.g. the data bits from ADDRESSa to ADDRESSa+(LENGTHa-1) or from ADDRESSa to ADDRESSa-(LENGTHa-1)), so that the data register 104 overwrites data stored in the memory units 106 corresponding to the physical memory addresses with the logic value ("1" or "0"). In addition, since the flag recorded in field 110a is "0" which means the data is not an end portion, the data clear module 100 keeps on reading a memory address ADDRESSb in field 108b, and clearing a plurality of data bits of a length LENGTHb according to a bit length LENGTHb recorded in field 112b. Again since the flag recorded in field 110b is "0", the data clear module 100 continues to clear data according to the memory address table 107. After the data clear module 100 reads a memory address ADDRESSn recorded in field 108n, and clears a plurality of data bits having a bit length LENGTHn recorded in field 112n, the

data clear module 100 stops reading memory addresses and clearing data bits since the flag recorded in field 110n is "1", which means the data is an end portion.

[0020] It is worth noting that in the above case the physical memory addresses are discontinuous, therefore the memory address table 107 is necessary for the data clear module 100 to clear data. If the physical memory addresses corresponding to the memory units 106 are continuous, the computer system 80 only has to provide a source memory address and the bit length of the data for the data clear module 100. The data clear module 100 can therefore consecutively clear data according to the source memory address and the bit length of the data.

[0021] Fig.4 is a schematic diagram of a computer system 120 according to a second preferred embodiment of the present invention. As shown in Fig.4, the computer system 120 includes a CPU 122, a north bridge circuit 124, a south bridge circuit 126, a display 128, a memory 130, an input device 132, and a hard disk 134. The north bridge circuit 124 includes a memory controller 136 and a display controller 138. The memory controller 136 includes a data clear module 140, an address register 142, and a data register 144. In this embodiment, the memory 130,

which comprises a plurality of memory units 150 arranged in arrays, is divided into a system memory 146 and a display memory 148.

[0022] Similar to the first embodiment, the data clear module 140 can clear data bits stored in the plurality of memory units 150 whether the corresponding physical memory addresses are continuous or discontinuous. If the physical memory addresses are discontinuous, the data clear module 140 can implement the clearing operation via a memory address table. If the physical memory addresses are continuous, the data clear module 140 only requires a source memory address and a bit length of the data to be clear, for clearing the data bits stored in the memory units 150 corresponding to the physical memory addresses.

[0023] It is to be noted that the data clear module 140 can also clear the memory units 150 of the display memory 148. Normally, the display controller 138 uses the display memory 148 to store operation data of 2D and 3D graphic calculations. The display memory 148 includes an image buffer zone and a Z buffer zone, where the image buffer is for storing the display data (e.g. gray scale value) of each pixel in the display 128, and the Z buffer is for storing the relative depth value of the display data in each

pixel. After the display controller 138 reads the display data stored in the image buffer to drive the display 128 for displaying an image, the display controller 138 must clear data stored in the image buffer and in the Z buffer before next image is displayed. In such case, the CPU 122 will output a logic value to the data clear module 140 once, and the memory addresses of the memory units 150 corresponding to the image buffer and the Z buffer to the address register 142. Meanwhile, the data clear module 140 starts to output a predetermined logic value ("1" or "0") to the data register 144 so that the data clear module 140 can overwrite the memory units 150 corresponding to the image buffer and the Z buffer with the predetermined logic value according to the memory addresses held in the address register 142.

[0024] Since the data clear module 140 can clear data of the display memory 148 without the involvement of the CPU 122, the CPU 122 can save more resources for other programs. In addition, since the limited bandwidth of the FSB between the CPU 122 and the north bridge circuit 124 is not consumed, the computer system 120 is more efficient.

[0025] Those skilled in the art will readily appreciate that numerous modifications and alterations of the device may be

made without departing from the scope of the present invention. Accordingly, the above disclosure should be construed as limited only by the metes and bounds of the appended claims.